Sistema para el Control, Administración y Programación de Robots Educativos

Germán Osella Massa[1], Cecilia De Vito[1], David Fernandez[1], Mónica Sarobe[1], Claudia Russo[1],

[1] Instituto de Investigación y Transferencia en Tecnología (ITT), Escuela de Tecnología, Universidad Nacional del Noroeste de la Provincia de Buenos Aires (UNNOBA). Sarmiento y Newbery (CP 6000), Junín, Buenos Aires, Argentina. Tel: (0236) 4636945/44
{german.osella, cecilia.devito, david.fernandez, monica.sarobe, claudia.russo}@itt.unnoba.edu.ar

Resumen: El presente trabajo trata sobre el sistema creado para el control y la programación del Robot Educativo Programable desarrollado en la UNNOBA, detallando diferentes casos de uso que permite la arquitectura de software propuesta y describiendo la biblioteca de funciones diseñadas para comandar al mencionado robot. Este sistema permite expandirse tanto para controlar nuevos robots así como para emular otras bibliotecas incluidas con otro tipo de robots.

Palabras clave: Robot, Educación, Programación, Administración.

1 Introducción

En la UNNOBA se está trabajando en el Robot Educativo Programable (REP), que combina la máxima integración de partes nacionales con la posibilidad concreta de ser armado y puesto en marcha por alumnos avanzados de escuelas con carreras técnicas, buscando brindar interfaces de programación acorde a las habilidades que se quieran desarrollar en los alumnos de los diversos niveles educativos.

El prototipo actual, una evolución del descripto en [1], posee un chasis de policarbonato en el que se montan dos motores de corriente continua con rodamientos de espuma de goma junto con una tercera rueda que actúa de apoyo. Cuenta con un procesador Arduino Nano, basado en un ATmega328 de 8 bits funcionando a 16 MHz, con 14 pines de E/S digital, 8 de E/S analógica, 30 KB de memoria flash, 2 KB de memoria SRAM y 1 KB de memoria EEPROM. Se agregan dos sensores ultrasónicos al frente del robot y dos sensores IR orientados hacia abajo para funcionar como detector de líneas. Cuenta con un módulo bluetooth de comunicación inalámbrica. La tecnología electrónica, el diseño industrial y la selección de materiales fue elegida por adaptarse a las posibilidades actuales de nuestro país.

El proyecto es un desarrollo libre y abierto. Las especificaciones técnicas, los esquemáticos de los circuitos y los diseños de varias piezas a producir mediante impresión 3D están disponibles bajo una licencia permisiva. El código fuente del software asociado se encuentra publicado en GitHub [2] bajo licencia GPL versión 3.

Este trabajo describe en detalle el sistema de software diseñado para la utilización del robot dentro del aula. En la sección 2 se presenta la arquitectura del sistema implementado, explorando distintas formas de emplearlo. En la sección 3 se detalla la interfaz de programación provista por la biblioteca usado por el usuario para controlar a los robots. En la sección 4 se ahonda en cuestiones técnicas relacionadas con el diseño interno del sistema. Finalmente, en la sección 5 se expondrán algunas conclusiones y futuros trabajos.

2 Arquitectura del sistema de control

Uno de los objetivos para el robot es que sea apto para la enseñanza, por lo que se consideró importante simplificar las tareas asociadas con su uso, particularmente cuando debe ser programado por usuarios inexpertos. El robot no es autónomo sino que se lo comanda desde una computadora en la que se realizan todas las tareas asociadas con la programación, utilizando órdenes simples dadas en un lenguaje de muy alto nivel. Esto evita que el usuario requiera conocimientos específicos de la arquitectura del hardware para poder cargar un programa en la memoria del microcontrolador que gobierna al robot. No obstante, esto demanda de algún tipo de comunicación inalámbrica entre el robot y la máquina que lo comanda. Por simplicidad operativa, se utiliza el estándar bluetooth, buscando el balance entre las variables costos/disponibilidad/confiabilidad. Se descartó el uso de transmisores RF y XBee por necesitar de hardware adicional específico y por los costos involucrados.

La comunicación serie provista por bluetooth es punto a punto. Si un programa establece una conexión con el robot, el enlace se produce de forma exclusiva, impidiendo que otros programas tengan acceso simultáneo y prohibiendo la creación de monitores del estado interno del robot o de mecanismos de parada de emergencia, fundamentales cuando se observa un comportamiento erróneo o peligroso.

Para sortear esta limitación, el sistema de control se vale de una arquitectura cliente/servidor en donde el servidor, denominado “Administrador”, actúa de nexo entre los robots y los programas escritos por los usuarios. Un administrador adquiere el control exclusivo de uno o más robots a través de sus respectivos enlaces bluetooth, brindando acceso a los mismos a todo cliente que se conecte con él a través de una red TCP/IP, creando así un nivel de indirección adicional entre el usuario y el robot, eludiendo de esta forma la limitación expuesta previamente.

Un programa del usuario se transforma entonces en un cliente de un administrador tras incorporar la biblioteca del sistema de control desarrollada para tal fin. Esta biblioteca contiene la lógica de comunicación con los administradores, al mismo tiempo que provee todos los comandos necesarios para controlar a los robots.

Cuando se está aprendiendo a programar es importante poder inspeccionar que está sucediendo mientras el programa recién escrito se está ejecutando, permitiendo así el desarrollo de modelos de pensamiento que expliquen los resultados observados. Un administrador incorpora la facilidad de generar notificaciones por cada comando enviado a un robot, junto con el resultado de ejecutarlo, habilitando así la creación de monitores que permitan explicar el comportamiento observado.

Para no limitar el uso del robot a un sistema operativo, tanto el servidor de control como la biblioteca para el usuario son multiplataforma, ejecutándose tanto sobre Windows como sobre sistemas basados en GNU/Linux o Mac OS X. La biblioteca también funciona sobre Android, permitiendo controlar al robot desde una tablet o teléfono inteligente. El código del sistema está escrito en Python 3 [3], siendo factible crear nuevas bibliotecas en otros lenguajes de programación.

El sistema es adaptable, pudiendo aplicarse tanto para controlar a un único robot comandado desde una sola computadora hasta facilitar la interacción simultanea con muchos robots controlados desde varias máquinas, en donde la cantidad de robots no es necesariamente la misma que la de computadoras controlándolos.

La progresión de casos de uso planteada para este sistema comienza con una única computadora sobre la cual se realiza toda la programación, controlando a uno o más robots mediante el uso de un adaptador bluetooth (Figura 1). Cada robot debe ser asociado una única vez con la computadora en cuestión, como cualquier otro dispositivo bluetooth, asignándole un nuevo puerto serie propio. Una vez hecha esta asociación, todo programa escrito para controlar a los robots deberá iniciarse indicando que se desea trabajar con un robot en particular, dándole un nombre que lo identifique junto con el puerto serie correspondiente. Esto debe repetirse para cada robot que se desea controlar. Tras registrar el primer robot, automáticamente se crea un administrador local oculto para interactuar entre el programa y los robots, encargándose de la comunicación bluetooth y del diálogo con los robots. Dicho administrador se detendrá automáticamente tras finalizar la ejecución del programa que lo inició. Este esquema es útil para realizar pruebas rápidas o cuando se cuenta con un robot por computadora, no siendo adecuado para compartir el uso de los robots entre varias máquinas.

Para esta segunda situación se requiere de un administrador explícitamente iniciado, actuando de nexo entre los robots y las demás computadoras o dispositivos desde donde controlarlos (Figura 2). Dicho administrador debe configurarse con los nombres de todos los robots a los que proveerá acceso, junto con los puertos de comunicación asociados a los enlaces por bluetooth. El administrador recibirá las órdenes a enviar a los robots, respondiendo a cada una de ellas con el resultado de su ejecución. También comunicará a todo cliente interesado información describiendo las órdenes ejecutadas junto con otros tipos de notificaciones relevantes. Este esquema resulta apropiado cuando se cuenta con varios robots y se los quiere compartir por turnos para controlarlos desde otras máquinas conectadas dentro de la misma red. Se requiere de una computadora donde ejecutar el administrador, la que igualmente puede usarse como puesto de trabajo adicional. Es obligatorio para este esquema disponer de una red de datos que comunique a todas las máquinas o dispositivos con aquella sobre la cual funcione el administrador.

Finalmente, es posible segregar a los robots en grupos controlados por diferentes administradores ejecutándose sobre máquinas distintas (Figura 3). Con este esquema se consigue separar físicamente a los robots, alejando a los administradores entre sí mucho más allá del radio de alcance típico del estándar bluetooth. También permite agrupar a ciertos robots por área temática, equipos o bandos. Un dispositivo podrá tener acceso a todos los robots disponibles en un administrador y también podrá comunicarse con varios administradores a la vez, teniendo así el potencial control de todos los robots que haya. Este esquema es el más complejo que se ha contemplado.

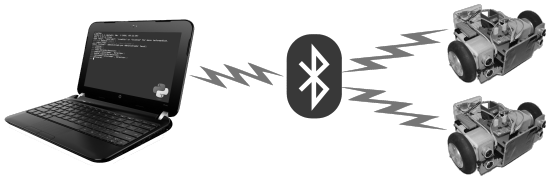

Figura 1. Uno o más robots controlados desde una única computadora vía bluetooth.

Figura 2. Muchos dispositivos controlando a varios robots a través de un Administrador.

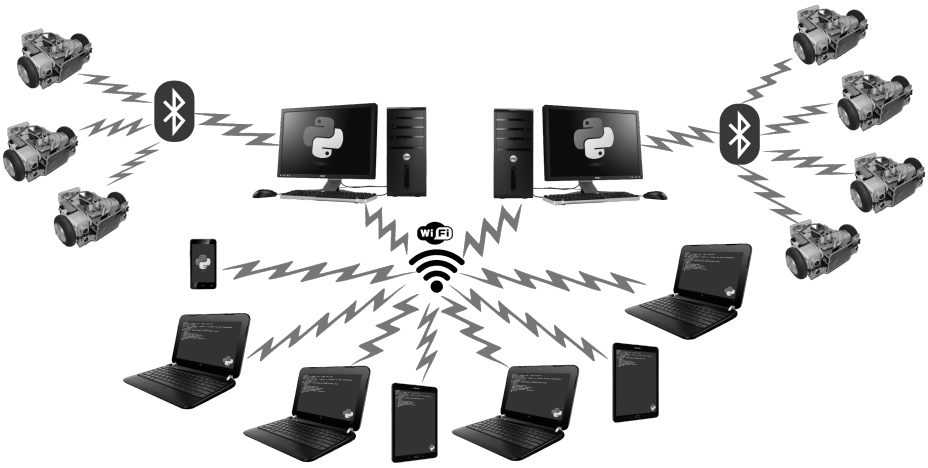

Figura 3. Administradores proveyendo a todos los dispositivos en red el acceso a los robots.

La arquitectura del sistema también es extensible, permitiendo desarrollar nuevos clientes que provean otras interfaces de programación para los robots y previendo la creación de nuevos administradores que comanden a otros tipos de robots. Como prueba de concepto, se desarrolló un módulo que emula la interfaz del DuinoBot de los N6 [4] utilizados en las experiencias llevadas a cabo por docentes investigadores del LINTI [5], traduciendo sus comandos en las correspondientes acciones sobre los robots de la UNNOBA. De esta forma, es posible reutilizar con mínimas modificaciones todos los programas ya escritos para los primeros.

3 Biblioteca para el control del robot

La biblioteca creada para controlar a los robots se diseñó para ser simple pero completa, potente y homogénea, buscando ser empleada con fines didácticos tanto en el aprendizaje de la programación imperativa así como en otras áreas de la currícula. Se eligió a Python 3 como lenguaje de programación utilizado para comandar al robot, por ser claro, predecible y aplicado con éxito en el ámbito académico [6]. Tanto la biblioteca de alto nivel como el administrador están escritos en este lenguaje.

La biblioteca del robot expone toda la funcionalidad necesaria para su uso, la que puede catalogarse en agregar y quitar robots locales y administradores remotos, solicitar a un robot que realice varios tipos de movimientos y encuestarlo por el estado de sus sensores. También incluye funciones auxiliares convenientes para escribir programas que controlen robots, incorporando manejo de tiempos, esperas y toma de decisiones al azar.

Los nombres de las funciones expuestas por la biblioteca están en español, escritos usando acentos y eñes donde corresponda, aprovechando el soporte nativo de Unicode [7] que Python 3 posee para los identificadores de un programa.

La biblioteca provee dos formas de ser utilizada, según el paradigma de programación que se desee emplear. Es posible usar una interfaz netamente procedural, la cual resulta sencilla cuando se trabaja con un único robot. También se provee una interfaz basada en objetos, que simplifica notablemente el control de varios robots simultáneamente. Una vez instalado el intérprete de Python 3 junto con el software para el control del robot, se sugiere utilizarla tanto en un programa como en el shell interactivo importándola directamente con `from edubots import *`

Este encantamiento mágico importa el módulo principal de la biblioteca del robot y trae al ámbito global del programa todas las funciones provistas por la misma. Si bien esta no es una práctica recomendable cuando se escriben programas grandes compuestos por varios módulos desarrollados en forma independiente por distintos programadores, resulta extremadamente conveniente cuando se intenta dar los primeros pasos introductorios, en donde los programas son relativamente pequeños y contenidos en un único módulo.

Los comandos disponibles tras importar la biblioteca pueden agruparse según la tarea asociada a los mismos. A continuación se describen los comandos asociados netamente con cuestiones administrativas, que permiten registrar robots para usarse directamente desde la máquina en la que se lo programa o para informar al sistema de la existencia de administradores que provean acceso a los robots:

`robot_local(nombre, dispositivo)`: Registra la existencia de un robot que se identificará con el nombre dado, comunicándose a través del dispositivo indicado. Esta función retorna un objeto que representa al robot recién registrado.

`olvidar_robot(nombre)`: Solicita al sistema la desactivación y olvido del robot identificado con el nombre dado. Tras olvidar exitosamente al robot, éste no podrá seguir usándose hasta tanto no se lo vuelva a registrar.

`administrador_remoto(dirección)`: Registra un administrador ubicado en la dirección IP dada como parámetro. Una vez agregado, todos los robots que ese administrador controle estarán disponibles para ser usados desde el programa. Retorna un objeto que representa al administrador registrado.

`olvidar_administrador(dirección)`: Quita un administrador previamente registrado con esa dirección. Los robots que ese administrador poseía desaparecerán.

`administradores()`: Devuelve una lista con los administrados registrados.

`robots()`: Devuelve una lista con los nombres de todos los robot conocidos.

`robot([nombre])`: Establece al robot identificado con el nombre dado para que sea el que ejecutará todos los sucesivos comandos. En caso que no se dé un nombre, se elige automáticamente al primer robot disponible. Retorna el robot elegido.

`nombre()`: Devuelve el nombre que identifica al robot usado por defecto.

Los siguientes comandos permiten realizar una interacción básica con un robot:

`suspender()`: Causa que el robot por defecto se detenga e ignore los sucesivos comandos recibidos, hasta tanto no se vuelva a reanudar su funcionamiento.

`reanudar()`: Reactiva al robot por defecto para que vuelva a aceptar comandos.

`detenerse()`: Para los motores del robot por defecto, deteniéndolo.

También se dispone de comandos asociados con el movimiento del robot, los que son simples, homogéneos y prevén un amplio abanico de movimientos pre-programados. No se describirán por separado a cada una de las funciones disponibles puesto que se espera su significado sea obvio. Las funciones con las que se cuenta son:

```
avanzar([velocidad], [duración])
retroceder([velocidad], [duración])
doblar_izquierda([velocidad], [duración])
doblar_derecha([velocidad], [duración])
doblar_izquierda_avanzando([velocidad], [duración])
doblar_derecha_avanzando([velocidad], [duración])
doblar_izquierda_retrocediendo([velocidad], [duración])
doblar_derecha_retrocediendo([velocidad], [duración])
rotar_izquierda([velocidad], [duración])
rotar_derecha([velocidad], [duración])
```

Todas las funciones efectúan la acción que su nombre refleja y, opcionalmente, la realizan a la velocidad dada y durante la duración indicada. La velocidad se especifica como un valor dentro del intervalo [0, 1], permitiendo así que sea expresada como valores con decimales (0,5 para moverse a “media máquina”) o como una fracción (siendo ¼, ½ o ¾ valores usuales). Si no se especifica la velocidad, se asume un desplazamiento a media potencia (es decir, a 0,5 o ½). La duración se expresa en segundos, dentro del intervalo [0, 60], pudiendo indicarse valores con parte fraccionaria (2,25 segundos, por ejemplo). En el caso que efectivamente se dé una duración, la acción tardará en ejecutarse exactamente el tiempo señalado, deteniendo el flujo normal de control del programa durante ese tiempo. Si en cambio no se da una duración, se asume que el movimiento será infinito y la ejecución del programa se reanuda tan pronto como el robot comience a moverse. Este manejo de la duración permite escribir fácilmente dos clases diferentes de programas: Aquellos que se asemejen a una coreografía se beneficiarán al especificar movimientos con duración, evitando tener que introducir pausas explícitas. Por otro lado, los que deben reaccionar instantáneamente a cambios en el ambiente necesitan tomar decisiones a partir de la información captada por los sensores al mismo tiempo que ajustar el desplazamiento que está realizando el robot, haciendo prohibitivo el uso de pausas.

La biblioteca provee una función de movimiento adicional que permite un control directo de las velocidades individuales de cada motor:

`moverse([izquierda],[derecha],[duración])`: Indica por separado la velocidad para los respectivos motores izquierdo y derecho del robot. La velocidad está expresada dentro del intervalo [-1, 1], donde un valor positivo indica una velocidad de avance mientras que un valor negativo marca una velocidad de retroceso. Una velocidad en cero detiene al motor y es el valor por defecto si no se la indica. La duración se interpreta igual que en los otros comandos de movimiento.

Para cubrir la necesidad de acceder al estado reportado por los sensores del robot, se decidió crear funciones específicas para cada tipo de sensor, dejando en evidencia el origen de la medición obtenida. El robot posee al frente dos sensores ultrasónicos ubicados en los laterales, que son capaces de detectar obstáculos hasta una distancia de 50 cm. Las siguientes funciones reportan sus mediciones:

`sensor_distancia(sensor)`: Retorna la distancia en centímetros al objeto más próximo (o 51 si no se detectó objeto alguno) usando el sensor indicado en el parámetro dado de la forma 'delantero-izquierdo' o 'delantero-derecho' (también se admiten los nombres más generales de 'izquierda' o 'derecha' dado que no hay sensores de proximidad traseros). Como cada medición consume cierto tiempo en realizarse, puede resultar más beneficioso obtener sólo el valor de un único sensor.

`sensores_distancia()`: Retorna una tupla con las distancias provistas por los sensores de proximidad. Es equivalente a invocar dos veces a la función anterior pero involucra un diálogo más corto con el robot.

El robot también posee dos sensores IR en su parte inferior. Como la medición de ambos sensores se realiza simultáneamente, se provee una única forma de obtenerlas:

`sensores_línea()`: Retorna una tupla con las mediciones provistas por los dos sensores IR. Los valores arrojados se encuentran en el rango [0, 1023].

Ciertas acciones aparecen en forma recurrente en los programas que controlan al robot. Si bien todas las funciones que se describirán a continuación pueden ser provistas por la biblioteca estándar de Python, para usarlas es necesario introducir los conceptos de espacios de nombres, módulos y como importarlos (o continuar con el uso de encantamientos mágicos, los que resulta preferible reducir o directamente eliminar). Para evitar esto, se proveen las siguientes funciones:

`esperar(duración)`: Detiene la ejecución del programa haciendo una pausa durante la duración indica, expresada en segundos como en los comandos anteriores.

`tiempo()`: Devuelve el tiempo actual expresado en segundos. Resulta principalmente útil para calcular el tiempo transcurrido haciendo la diferencia entre los resultados obtenidos en dos invocaciones diferentes a esta función.

`al_azar([probabilidad=50])`: Devuelve verdadero un porcentaje de las veces que se la invoca, en forma acorde a la probabilidad dada, expresada en el rango [0, 100]. Si no se indica una probabilidad, retorna verdadero el 50% de las veces.

Todas las funciones descriptas fueron presentadas en forma procedural, donde los comandos se traducen en órdenes enviadas a un robot predefinido que, por defecto, es el primer robot local que se haya registrado o, si no hubiera alguno local, será el primer robot disponible en el primer administrador conocido. Si tampoco se hubiera configurado al menos un administrador, la ejecución de cualquier comando para el robot levantará una excepción señalando este problema.

Para controlar a más de un robot a la vez, la biblioteca provee la función `robot()` que, tras su invocación exitosa, cambia el robot al que implícitamente se envían los siguientes comandos. El robot seleccionado forma parte del estado global del programa pero no se tiene acceso explícito para modificarlo.

La función `robot()` también es la puerta de acceso al enfoque orientado a objetos provisto por la biblioteca ya que retorna un objeto que representa al robot identificado con el nombre dado como argumento (sin importar si el robot fuera local o remoto). Dicho objeto provee a través de sus métodos y atributos toda la funcionalidad detallada anteriormente, con exactamente los mismos nombres y parámetros que su contrapartida procedural. El siguiente código contrasta lado a lado el mismo programa escrito usando el paradigma procedural (izquierda) y el orientado a objetos (derecha):

```
...                                  |  ...
robot('Uniqua')                      |  uniqua = robot('Uniqua')
avanzar(duración=1.5)                |  uniqua.avanzar(duración=1.5)
esperar(2)                           |  esperar(2)
rotar_derecha(duración=4.5)          |  uniqua.rotar_derecha(duración=4.5)
...                                  |  ...
```

El enfoque orientado a objetos requiere del uso explícito del objeto que representa al robot, volviéndolo más largo de escribir. Desde el punto de vista del aprendizaje, involucra más conceptos que el procedural, en donde no se necesita hablar de objetos ni explicar la diferencia entre invocar a un método e invocar a una función.

En contrapartida, cuando en un programa se debe controlar a más de un robot a la vez, el enfoque orientado a objetos resulta más simple de razonar, dado que cada comando indica explícitamente a qué robot va dirigido. En el enfoque procedural, los comandos son enviados al último robot seleccionado, que forma parte del estado global implícito del programa, causando que no sea obvio quién los ejecutará. Esto se ve agravado cuando se debe cambiar constantemente de robot para poder intercalar el envío de órdenes a cada uno de ellos: No es posible determinar con qué robot se está trabajando en cada momento sin seguir el código de principio a fin. Cabe remarcar que si se trabaja con un único robot, la cuestión anterior pierde importancia dado que como el robot es siempre el mismo, se lo puede considerar efectivamente como una constante global e ignorarlo en la interpretación de los comandos.

4 Detalles de implementación del sistema de control

La comunicación por bluetooth entre el administrador y los robots es regida por un protocolo propio desarrollado específicamente con el objetivo de ser simple y austero pero cubriendo todas las necesidades actuales, previendo ser extendido en el futuro si se amplían las capacidades del robot. Codifica cuatro comandos fundamentales:

- Arranque: Tras encenderse, el robot se inicia ignorando todos los comandos recibidos excepto éste, que lo habilita para actual ante el resto de los comandos.
- Finalización: Detiene los motores del robot y lo vuelve al estado inicial.

- Movimiento: Indica con qué velocidad se debe accionar a cada rueda del robot. Las velocidades se dan en el rango [-255, 255] donde un valor positivo hace que la rueda avance mientras que uno negativo la hace retroceder y cero la detiene. El comando incluye una duración en el rango [0, 60000] representando el tiempo en mili-segundos durante el cual se lleva a cabo el movimiento. Se admite el valor especial 65535 (o FFFFh) que representa una duración infinita.
- Sensores: Encuesta el estado de los sensores del robot. Es posible recibir en la respuesta información sobre todos o sólo un subconjunto de los sensores.

Los comandos de movimiento y de sensores son respondidos inmediatamente tras ser recibidos, confirmando la recepción exitosa. Adicionalmente, para los comandos con duración finita se envía un segundo mensaje cuando termina su ejecución.

Para implementar la comunicación entre el administrador y sus clientes se utiliza la biblioteca ØMQ [8], que provee una abstracción sobre los sockets de TCP/IP, facilitando la creación de protocolos de alto nivel y simplificando la tarea de implementar distintos patrones de comunicación. La información intercambiada a través de los sockets es codificada en formato JSON [9] para que sea fácilmente generada y/o consumida desde distintos dispositivos o lenguajes de programación.

Cuando el administrador entra en actividad, éste abrirá dos sockets ØMQ para interactuar con los potenciales clientes: El primero, ubicado en el puerto 7060, emplea un modelo de comunicación de tipo REQ/REP de ØMQ (de pedido y respuesta), donde se recibe un pedido asociado con la administración de los robots para luego enviarse la respuesta con el resultado antes de volver a atenderse el siguiente pedido.

Los pedidos reconocidos son: Obtener los nombres de los robots, solicitar el acceso a un robot en particular, suspender el envío de comandos a un robot, deteniéndolo y reanudar el envío de comandos a un robot previamente suspendido.

El administrador abre adicionalmente un segundo socket ØMQ ubicado en el puerto 7061, que emplea un modelo de comunicación del tipo PUB/SUB de ØMQ (de publicación y subscripción). A través de este socket se distribuyen notificaciones sobre cada comandos ejecutado junto con su resultado. El propósito de este socket es diseminar toda la actividad que fluye a través del administrador, con el fin de ser usada por programas monitores o de depuración. El modelo PUB/SUB permite subscribirse para recibir todas o sólo un subconjunto de las posibles notificaciones.

Cuando se solicita el acceso a un robot, el administrador abre un nuevo socket ØMQ ubicado en un puerto al azar, informándoselo al cliente que desea controlarlo. A través de ese socket de carácter transitorio, que emplea un patrón de comunicación REQ/REP, es que la biblioteca finalmente puede enviar órdenes al robot. Las órdenes tienen una correspondencia directa con las instrucciones que se describieron en la sección 3. Actualmente se emplea una política permisiva con respecto al envío de comandos, aceptando que dos o más clientes envíen comandos simultáneamente al mismo robot. Los comandos serán ejecutados en un orden secuencial FIFO, es decir, que el primero en llegar será el primero en ser atendido, pero no habrá superposición en la ejecución de los mismos. Esto puede traer consecuencias no deseadas si se ejecutan dos o más programas simultáneamente, ya que probablemente ninguno de ellos logrará el comportamiento esperado.

5 Conclusiones y trabajo futuro

El sistema presentado en este artículo permite lograr el control de robots diseñados con fin educativo utilizando una interfaz simple y uniforme, tanto desde un enfoque procedural como desde uno orientado a objetos. Oculta al usuario todos los detalles relacionados con la comunicación inalámbrica y la coordinación con varios robots simultáneamente. Permite además que la organización que posea los robots elija la mejor forma de utilizarlos, adaptándose a las posibilidades y necesidades de la misma.

Como trabajo a futuro, se desea validar el uso del sistema en situaciones complejas, para detectar posibles debilidades que pudieran surgir en la arquitectura desarrolla. Se estudiará si existe la necesidad de incorporar algún mecanismo de autenticación y seguridad al sistema, usándolo para otorgar o revocar el uso exclusivo de los robots a determinados usuarios durante cierto tiempo, en contrapartida al uso actual más libre y permisivo aunque también más anárquico o necesitando de una coordinación ad-hoc. También resulta de interés crear ambientes integrados de desarrollo específicos para el robot junto con la exploración de otros lenguajes de programación desde los cuales controlar al robot, analizando sus fortalezas y debilidades comparándolas con el actualmente elegido. Se está comenzando a trabajar en la aplicación de lenguajes gráficos visuales basados en bloques como lenguaje alternativo, proveyendo además un puente de traducción automática entre los bloques visuales y el código textual de Python 3. Finalmente, queda planteado el desafío de estudiar la ampliación de la arquitectura para que la administración y control de los robots sea a través de internet, incorporándolos a un laboratorio virtual.

Referencias

1. Osella Massa, G., Álvarez, E., Useglio, G., Luengo, P., Llanos, E., Sarobe, M., Russo, C.: Programando y Pensando Robots en la Universidad y en Escuelas Secundarias. En el 1er. Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI 2013).
2. Repositorio conteniendo el código fuente del proyecto: https://github.com/gosella/edubots
3. Guido van Rossum et al, “The Python Language Reference”, Python Software Foundation; https://docs.python.org/3/reference/index.html
4. Programando con robots, LINTI, UNLP, http://robots.linti.unlp.edu.ar/
5. Díaz, F.J., Banchoff, C.M., Martin, E.S., & López, F.: Aprendiendo a programar con juegos y robots. VII Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET 2012).
6. Guo, P.: Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities. BLOG@CACM, Comunications of the ACM: http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/
7. El estándar de codificación de caracteres del Consorcio Unicode: http://unicode.org/
8. ØMQ Distributed Messaging. The ZeroMQ Reference Manual: http://api.zeromq.org/
9. JSON (JavaScript Object Notation) data-interchange format: http://www.json.org/